CISC105 Fall 2005 Lab07

- Solve each of the following problems. Be sure to save every separate program. All programs must be properly commented and indented (see Assignment Standards on the class website).
- Functions must be commented, but the names of functions and the names of parameters should clearly indicate their purpose. Comments are explanatory, names are informational.
- We have been working on file input. Please review your notes and your text (which has lots of good examples). You have also been learning to read man pages. How many have you read? Have you read the man pages for commands you already know how to use? What grade do you expect to get in this course?
- Even though these programs are simple and short, it would be smart to code them a few lines at a time, compile, test. Remember that printf is your best friend when it comes to debugging.
- **EVERY** file you open must have a conditional to test if it opened correctly, as shown in class.
- **EVERY** file you open must also be closed.
- Name each program lab11.n.c, where n is the number in the list below. For example, the name of the file for the first will be lab11.1.c

Programs

1. For each statement, show the effect by writing what is printed, drawing the effect on the stack, or writing ”error”.

```
int x = 7, y;
int *ptr1, *ptr2;

ptr1 = &x;
ptr2 = ptr1;

printf("%p",ptr1);

printf("%d\n", *ptr1);

printf("%p\n" ptr2,);

printf("%d\n", *x);

printf("%d\n", *&x);
```

2. For the following declarations, draw a picture as we do in class so you can answer the questions. Make up numbers for addresses, then use them consistently.

```
int x = 5;
int *ptr = &x;
```

```
What are the values of the following expressions?

     a. ptr

     b. *ptr

     d. ptr == &x

     e. &ptr

     f. *x

     g. *&x  //what can you say about this ordered pair of operators, *&?

     h. **&ptr
```

2. Create a data file using Emacs. Put three integers into it, separated by carriage returns (note: do not put "`\n`" into your file, just hit the return key). Be sure to follow the last integer with a single carriage return.

   Follow these instructions carefully. Do not add steps. In your main():

   (a) Declare a FILE pointer named **input**. Open your data file in your main() using fopen (see the man page if you don't remember what it takes as arguments or what it returns).

   (b) Check your FILE pointer with a conditional to be sure the file opened correctly, and print a nice error message if it doesn't open correctly. (Don't exit the program, though!).

   (c) Print the value of **input** as a pointer.

   (d) Print the value of **NULL** as a pointer.

   (e) Close your data file using fclose. Yadda yadda man page yadda textbook yadda testing.

   (f) Still in main(), and still using **input**, use fopen to open another filename that does *not* exist.

   (g) Check your FILE pointer with a conditional to see if the file opened correctly, and print a nice error message if it doesn't open correctly.

   (h) Print the value of **input** as a pointer.

   (i) Print the value of **NULL** as a pointer.

   (j) Close the FILE pointer, even though you don't think it worked. It's a good habit.

   (k) What have you demonstrated about FILE pointers and fopen? Think about: why do we use the constant NULL instead of the value it represents?

3. In a new program, open the data file you created in 2. Use three fscanf calls to read the three integers, and then print each integer out.

4. In main():

   (a) Declare an array of char of size 80 called **line**.

   (b) Declare a variable of type `char *` named "fgResult". Note that this is not the same as type `char`.

(c) Open your data file.

(d) Use fgets to get a line from your file and put it in **line**. Save the return value of your call to fgets in "fgResult". Print "fgResult" as a pointer, along with **line** (which is a what?).

(e) After that is working correctly, set up a loop to repeat part 4d a total of four times; once for each integer, and then once more. Observe what gets printed the fourth time carefully. Note in your comments what happens when fgets reaches the end of your data.

5. Write a loop that reads your data file, printing the value each time, until fgets returns NULL. (What kind of loop? How will you know when fgets returns NULL?). Add a counter to your loop so that the line number gets printed along with your data.

You should have a total of 5 programs named lab11.1.c to lab11.5.c. Make a single script file (see lab00 for the instructions) where you cat, rat, and elephant each program in its final form.

Submit all code and script files to /dev/null (ask your TA for assistance), and give the paper version of the **complete script file only** to the recycle bin at the beginning of your next lab. Note: Cat, compile, and run each program in order! Who knows what could happen otherwise? It is not to be contemplated. Do *not* cat all programs, then compile, etc.